

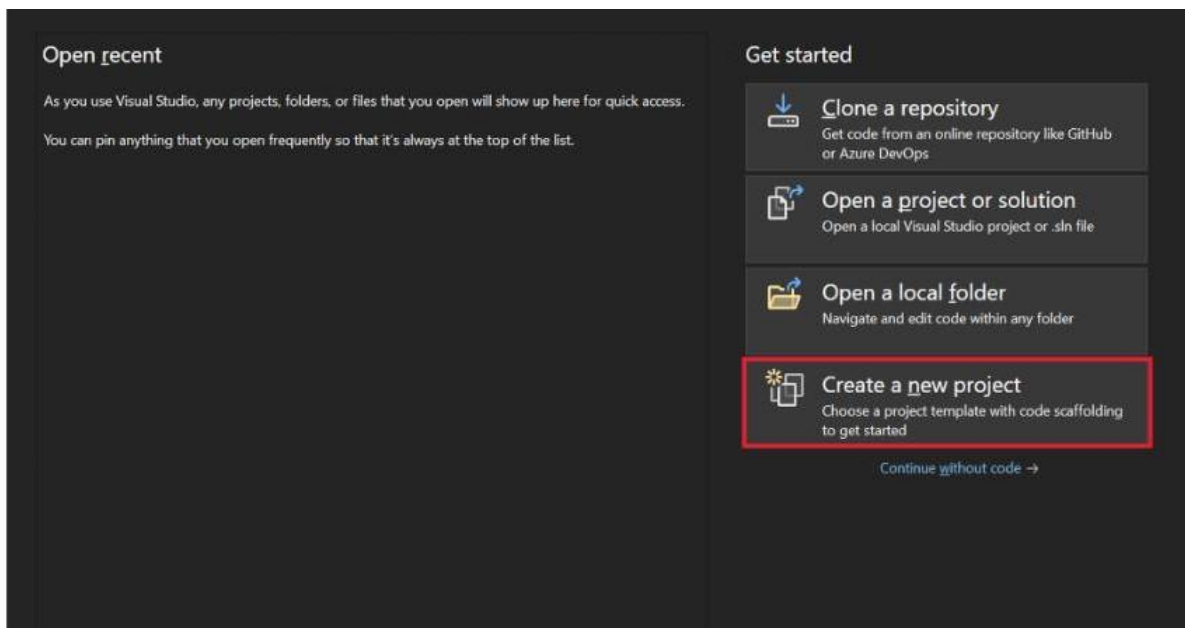
EPAM-LAB_10

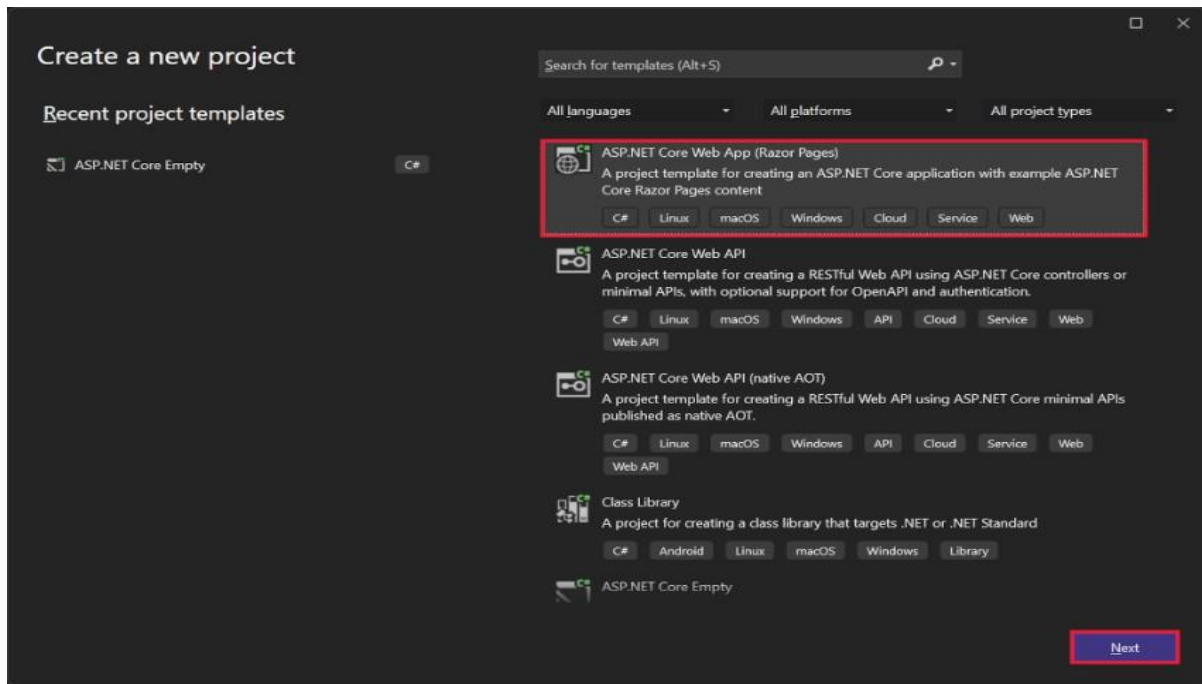
(Asp.Net using State management Technique)

Implement a website for Chatting using Asp.net web forms. By taking the following requirements.

- i) Create one login form with 1 label, 1 TextBox and 1 button
- ii) Create one ChatPage.html which will be open when user clicks on login button
- iii) Create Message.aspx for sending user given messages to the Application Object
- iv) Create Display.aspx for displaying all the messages in Chat Area from the Application object

Hint: Before sending messages to the Application, you must store Some messages in the Application object within Global.asax file. And also you must store UserName in the Session also.

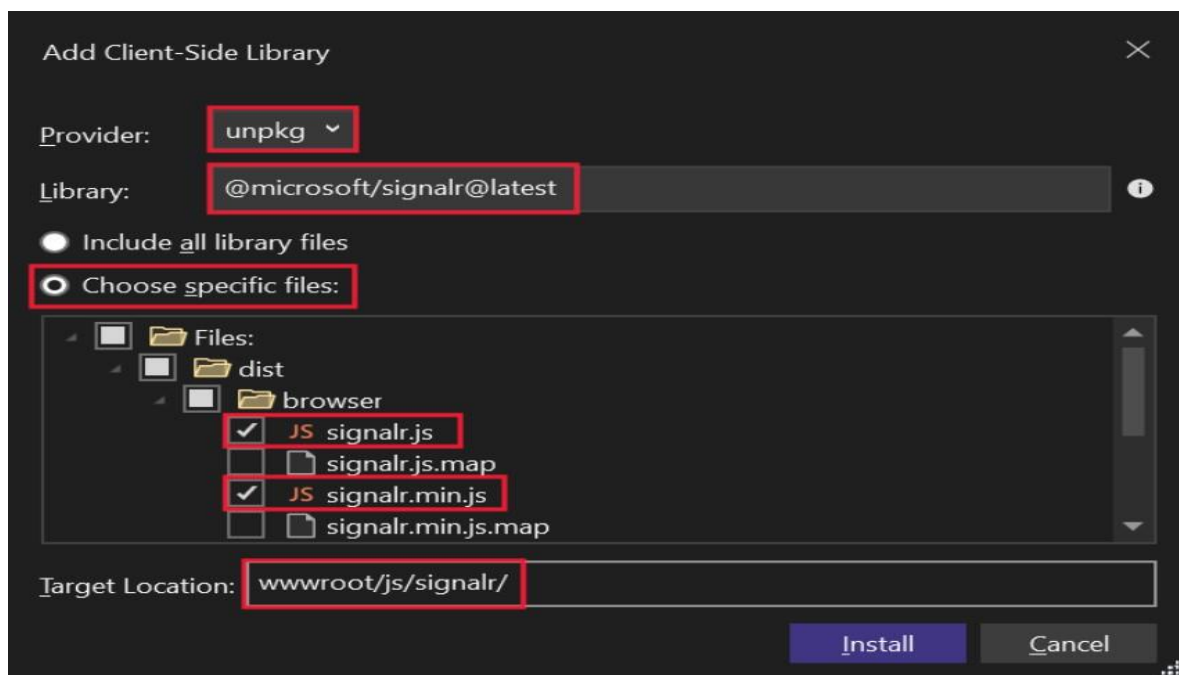




In **Solution Explorer**, right-click the project, and select **Add** > **Client-Side Library**.

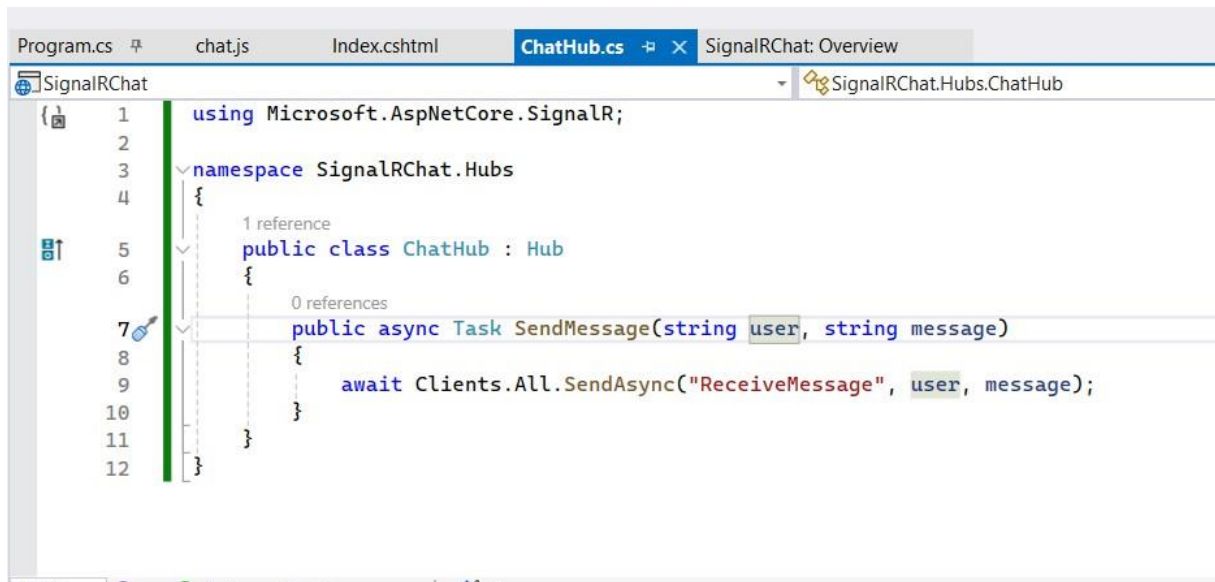
In the **Add Client-Side Library** dialog:

- Select **unpkg** for **Provider**
- Enter `@microsoft/signalr@latest` for **Library**.
- Select **Choose specific files**, expand the *dist/browser* folder, and select `signalr.js` and `signalr.min.js`.
- Set **Target Location** to `wwwroot/js/signalr/`.
- Select **Install**.



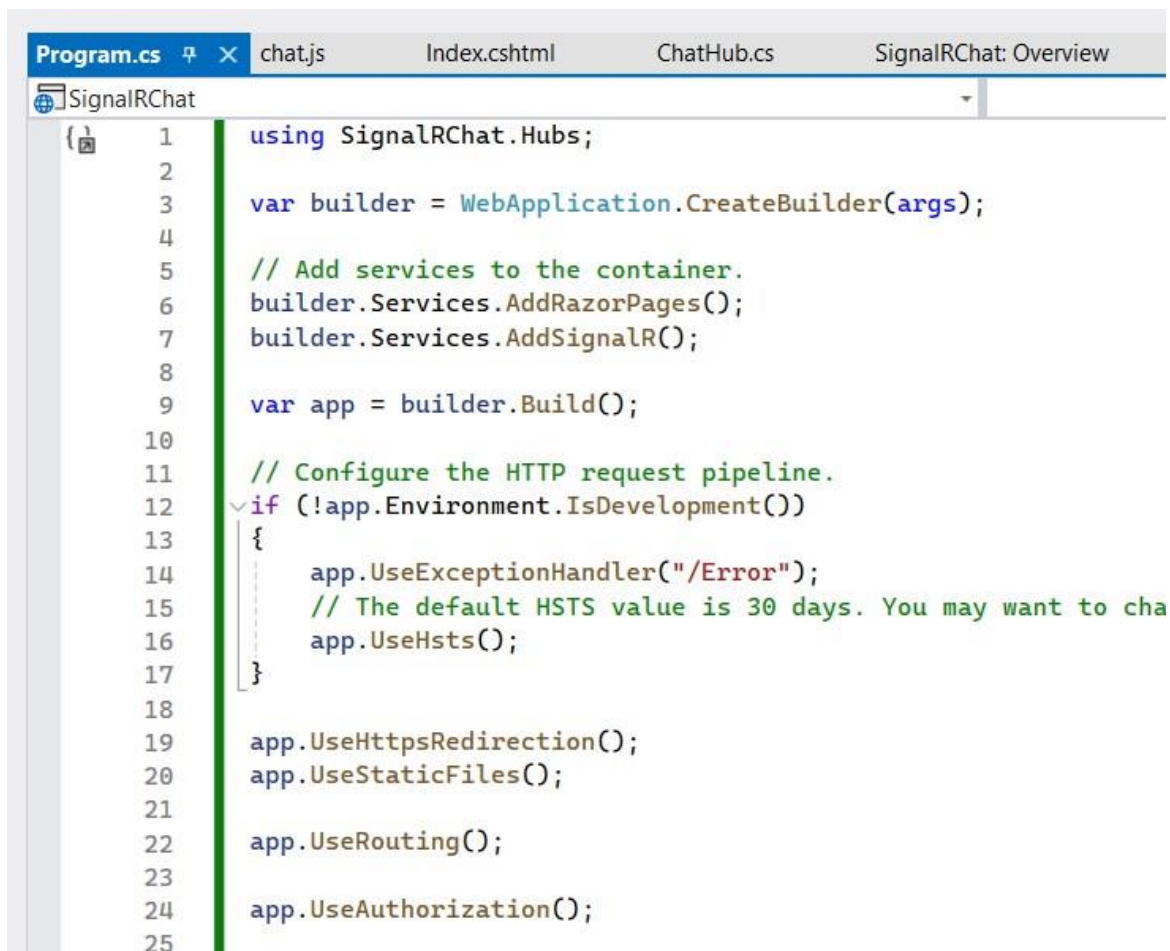
In the SignalRChat project folder, create a Hubs folder.

In the Hubs folder, create the chatHub class with the following code:



```
1 using Microsoft.AspNetCore.SignalR;
2
3 namespace SignalRChat.Hubs
4 {
5     public class ChatHub : Hub
6     {
7         public async Task SendMessage(string user, string message)
8         {
9             await Clients.All.SendAsync("ReceiveMessage", user, message);
10        }
11    }
12 }
```

The SignalR server must be configured to pass SignalR requests to SignalR. Add the following code to the Program.cs file.



```
1 using SignalRChat.Hubs;
2
3 var builder = WebApplication.CreateBuilder(args);
4
5 // Add services to the container.
6 builder.Services.AddRazorPages();
7 builder.Services.AddSignalR();
8
9 var app = builder.Build();
10
11 // Configure the HTTP request pipeline.
12 if (!app.Environment.IsDevelopment())
13 {
14     app.UseExceptionHandler("/Error");
15     // The default HSTS value is 30 days. You may want to cha
16     app.UseHsts();
17 }
18
19 app.UseHttpsRedirection();
20 app.UseStaticFiles();
21
22 app.UseRouting();
23
24 app.UseAuthorization();
25
```

```

25
26 app.MapRazorPages();
27 app.MapHub<ChatHub>("/chatHub");
28
29 app.Run();

```

Replace the content in Pages/Index.cshtml with the following code:

```

SignalRChat
1 @page
2 <div class="container">
3 <div class="row p-1">
4 <div class="col-1">User</div>
5 <div class="col-5"><input type="text" id="userInput" /></div>
6 </div>
7 <div class="row p-1">
8 <div class="col-1">Message</div>
9 <div class="col-5"><input type="text" class="w-100" id="messageInput" /></div>
10 </div>
11 <div class="row p-1">
12 <div class="col-6 text-end">
13 <input type="button" id="sendButton" value="Send Message" />
14 </div>
15 </div>
16 <div class="row p-1">
17 <div class="col-6">
18 <hr />
19 </div>
20 </div>
21 <div class="row p-1">
22 <div class="col-6">
23 <ul id="messagesList"></ul>
24 </div>
25 </div>
26 </div>
27 <div class="row p-1">
28 <div class="col-6">
29 <ul id="messagesList"></ul>
30 </div>
31 </div>
32 </div>
33 <script src="~/js/signalr/dist/browser/signalr.js"></script>
34 <script src="~/js/chat.js"></script>

```

In the wwwroot/js folder, create a chat.js file with the following code:

```
Program.cs  chat.js  Index.cshtml  ChatHub.cs  SignalRChat: Overview
SignalRChat JavaScript Content Files  connection.on("ReceiveMessage") callback

1  "use strict";
2
3  var connection = new signalR.HubConnectionBuilder().withUrl("/chatHub").build();
4
5  //Disable the send button until connection is established.
6  document.getElementById("sendButton").disabled = true;
7
8  4 references
9  connection.on("ReceiveMessage", function (user, message) {
10     var li = document.createElement("li");
11     document.getElementById("messagesList").appendChild(li);
12     // We can assign user-supplied strings to an element's textContent because it
13     // is not interpreted as markup. If you're assigning in any other way, you
14     // should be aware of possible script injection concerns.
15     li.textContent = `${user} says ${message}`;
16 }
17
18 4 references
19 connection.start().then(function () {
20     document.getElementById("sendButton").disabled = false;
21
22     0 references
23     }).catch(function (err) {
24         return console.error(err.toString());
25     });
26
27 122 references
28 document.getElementById("sendButton").addEventListener("click", function (event) {
29     var user = document.getElementById("userInput").value;
30     var message = document.getElementById("messageInput").value;
31
32     4 references
33     connection.invoke("SendMessage", user, message).catch(function (err) {
34         return console.error(err.toString());
35     });
36     event.preventDefault();
37 }
38 }
```

Run the app

Copy the URL from the address bar, open another browser instance or tab, and paste the URL in the address bar.

Choose either browser, enter a name and message, and select the **Send Message** button.

The name and message are displayed on both pages instantly.

Output:

